intel®

Low Voltage Intel® Xeon™ Processor

Specification Update

November 2002

Notice: The Low Voltage Intel® Xeon™ Processor may contain design defects or errors known as errata that may cause the product to deviate from published specifications. Current characterized errata are documented in this specification update.

.

Order Number: 273767-003

Contents

Revision History 4

Preface 5

Affected Documents/Related Documents 5
Nomenclature 6

Summary Table of Changes 7

Codes Used in Summary Table 7
Errata 9
Specification Changes 10
Specification Clarifications 10
Documentation Changes 10

Identification Information 11

Examples of Low Voltage Intel® Xeon™ Processor Markings 11
Examples of Production Markings 12
Identification Information 13

Errata 14

Specification Changes 23

Specification Clarifications 24

Documentation Changes 25

Revision History

Date	Version	Description
November 2002	003	Added Errata W25. Added Documentation Changes 1-10. Added Specification Clarification W2.
October 2002	002	Reformatted document.
September 2002	001	Initial release of the document.

Preface

This document is an update to the specifications contained in the Affected Documents/Related Documents table below. This document is a compilation of device and documentation errata, specification clarifications and changes. It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools.

Information types defined in Nomenclature are consolidated into the specification update and are no longer published in other documents.

This document may also contain information that was not previously published.

Affected Documents/Related Documents

Title	Order
Low Voltage Intel® Xeon™ Processor Datasheet at 1.60 GHz	273766
Intel Architecture Software Developer's Manual, *Volumes 1, 2 and 3* **NOTE: Starting next month, documentation changes for *IA-32 Intel® Architecture Software Developer's Manual* volumes 1, 2, and 3 will be posted in a separate document *IA-32 Intel® Architecture Software Developer's Manual Documentation Changes*. Please follow the link below to become familiar with this file.** http://developer.intel.com/design/pentium4/specupdt/252046.htm	245470, 245471, and 245472

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics, e.g., core speed, L2 cache size, package type, etc. as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number.

Errata are design defects or errors. These may cause the Low Voltage Intel® Xeon™ Processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices unless otherwise noted.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in any new release of the specification.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in any new release of the specification.

Documentation Changes include typos, errors, or omissions from the current published specifications. These will be incorporated in any new release of the specification.

Note: Errata remain in the specification update throughout the product's lifecycle, or until a particular stepping is no longer commercially available. Under these circumstances, errata removed from the specification update are archived and available upon request. Specification changes, specification clarifications and documentation changes are removed from the specification update when the appropriate changes are made to the appropriate product specification or user documentation (datasheets, manuals, etc.).

Summary Table of Changes

The following table indicates the errata, specification changes, specification clarifications, or documentation changes which apply to the Low Voltage Intel® Xeon™ processor product. Intel may fix some of the errata in a future stepping of the component, and account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

Codes Used in Summary Table

Stepping

X:	Errata exists in the stepping indicated. Specification Change or Clarification that applies to this stepping.
(No mark)	
or (Blank box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

Page

(Page):	Page location of item in this document.

Status

Doc:	Document change or update will be implemented.
Plan fix:	This erratum may be fixed in a future stepping of the component.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.

Row

|	Change bar to left of table row indicates this erratum is either new or modified from the previous version of the document.

Each Specification Update item will be prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Intel® Pentium® II processor
B = Mobile Intel® Pentium® II processor
C = Intel® Celeron® processor
D = Intel® Pentium® II Xeon™ processor
E = Intel® Pentium® III processor
G = Intel® Pentium® III Xeon™ processor
H = Mobile Intel® Celeron® processor at 466 MHz, 433 MHz, 400 MHz, 366 MHz, 333 MHz, 300 MHz, and 266 MHz
K = Mobile Intel® Pentium® III Processor - M
M = Mobile Intel® Celeron® processor
N = Intel® Pentium® 4 processor
O = Intel® Xeon™ processor MP
P = Intel® Xeon™ processor and Intel® Xeon™ processor with 512 KB L2 Cache
T = Mobile Intel® Pentium® 4 processor
V = Mobile Intel® Celeron® processor on .13 Micron Process in Micro-FCPGA Package
W = Low Voltage Intel® Xeon™ processor

The Specification Updates for the Intel® Pentium® processor, Intel® Pentium® Pro processor, and other Intel products do not use this convention.

Errata

No.	Steppings	Page	Status	ERRATA
	C1/0F27h			
W1	X	14	No Fix	Transaction is not Retried after BINIT#
W2	X	14	No Fix	Invalid Opcode 0FFFh Requires a ModRM Byte
W3	X	14	No Fix	When in No-Fill Mode (CR0.CD=1) the Memory Type of Large (PSE-4M and PAE-2M) Pages are Wrongly Forced to Uncacheable
W4	X	14	No Fix	Processor may Hang due to Speculative Page Walks to Non-Existent System Memory
W5	X	15	No Fix	Memory Type of the Load Lock Different from its Corresponding Store Unlock
W6	X	15	No Fix	Machine Check Architecture Error Reporting and Recovery may not work as Expected
W7	X	17	No Fix	Debug Mechanisms may not Function as Expected
W8	X	18	No Fix	Cascading of Performance Counters does not work Correctly when Forced Overflow is Enabled
W9	X	18	No Fix	EMON Event Counting of x87 Loads may not work as Expected
W10	X	18	No Fix	System Bus Interrupt Messages without Data and which Receive a HardFailure Response may hang the Processor
W11	X	19	No Fix	Processor Flags #PF Instead of #AC on an Unlocked CMPXC8B Instruction
W12	X	19	No Fix	FSW may not be Completely Restored after Page Fault on FRSTOR or FLDENV Instructions
W13	X	19	No Fix	Processor Issues Inconsistent Transaction Size Attributes for Locked Operation
W14	X	19	No Fix	IA32_MC0_ADDR and IA32_MC0_MISC Registers will Contain Invalid or Stale Data Following a Data, Address, or Response Parity Error
W15	X	20	No Fix	When the Processor is in the System Management Mode (SMM), Debug Registers may be Fully Writeable
W16	X	20	No Fix	Associated Counting Logic must be Configured when using Event Selection Control (ESCR) MSR
W17	X	20	No Fix	Shutdown and IERR# may Result due to a Machine Check Exception on a Hyper-Threading Technology Enabled Processor
W18	X	20	Plan Fix	BPM[5:3]# VIL does not meet Specification
W19	X	21	No Fix	Processor may hang under Certain Frequencies and 12.5% STPCLK# Duty Cycle
W20	X	21	No Fix	System may hang if a Fatal Cache Error causes Bus Write Line (BWL) Transaction to occur to the same Cache Line Address as an Outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)
W21	X	21	Plan Fix	Re-mapping the APIC Base Address to a Value Less Than or Equal to 0xDC001000 may cause IO and Special Cycle Failure
W22	X	21	Plan Fix	Erroneous BIST Result Found in EAX Register after Reset
W23	X	22	No Fix	Simultaneous Assertion of A20M# and INIT# may Result in Incorrect Data Fetch
W24	X	22	Plan Fix	Processor does not Respond to Break Requests from ITP
W25	X	22	Plan Fix	A Write to APIC Task Priority Register (TPR) That Lowers Priority May Appear to Have Not Occurred

Specification Changes

No.	Steppings	SPECIFICATION CHANGES
	C1/0F27h	
		None for this revision of this specification update.

Specification Clarifications

No.	Steppings	SPECIFICATION CLARIFICATIONS
	C1/0F27h	
W1	X	AE28 and AE29 Pins Signal Definitions are not VCC

Documentation Changes

No.	Page	DOCUMENTATION CHANGES
1	25	Inaccurate Operation Description for Shift Instructions
2	25	Documentation Changes to Support the Use of Hyper-Threading Technology
3	25	IRET/IRETD TASK-RETURN Block has Incorrect Exception Information
4	26	LGDT Exception Listing in Virtual-8086
5	26	Various Table B-22 Errors and Omissions
6	28	Misreporting of Exceptions for MOVDQA
7	29	Effect of STI on NMI
8	29	133 MHz PSB Encoding in MSR_EBC_FREQUENCY_ID Register
9	30	Double Fault Exception
10	30	Move To/From Control Registers

Identification Information

Examples of Low Voltage Intel® Xeon™ Processor Markings (604-pin FC-mPGA2 Package)

Figure 1. Top Side Processor Marking-Production Part

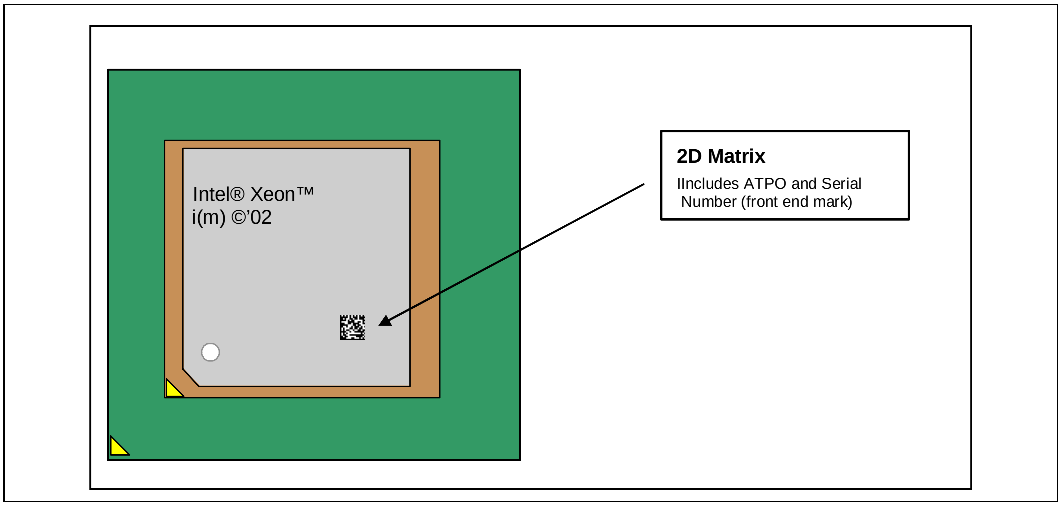
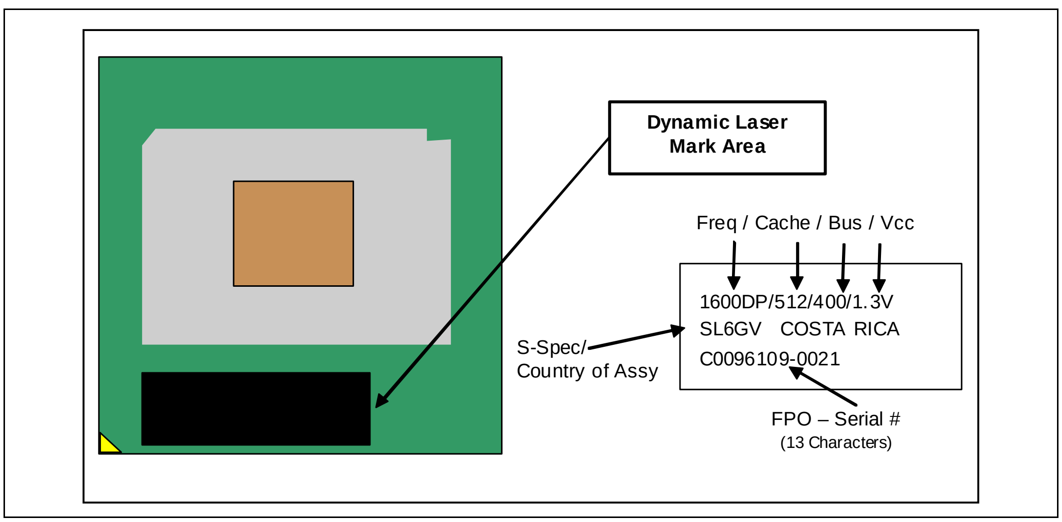


Figure 2. Bottom Side Processor Marking



Examples of Production Markings

Figure 3. Top View

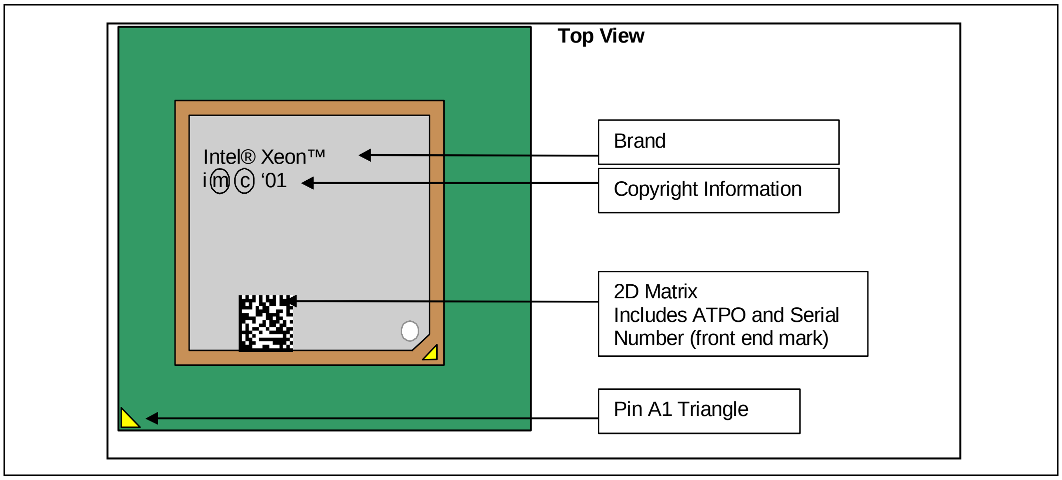
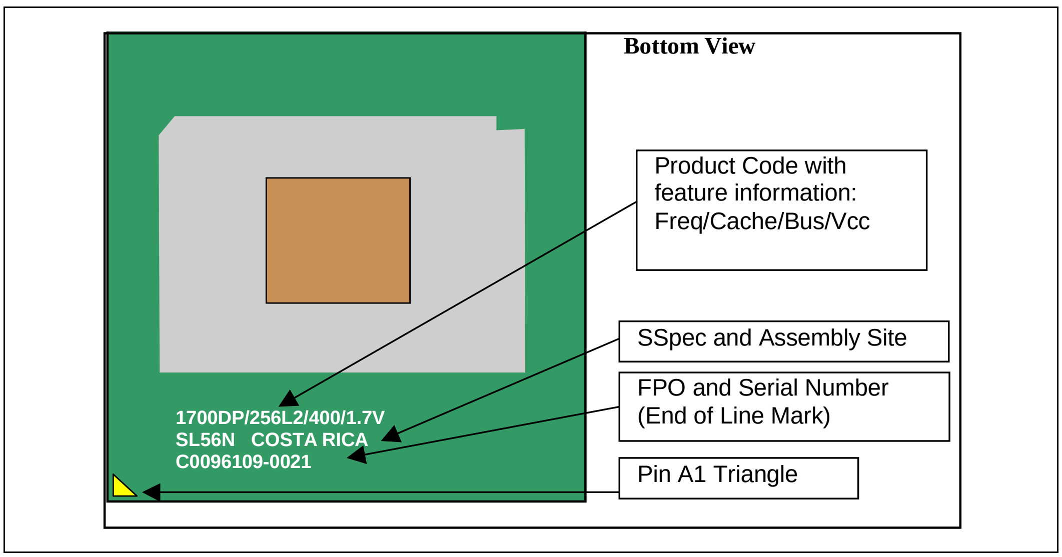


Figure 4. Bottom View



Identification Information

The Low Voltage Intel® Xeon™ processor may be identified by the following values:

Family[1]	Model[2]	Brand ID[3]
1111	0010	00001011

NOTES:
1. The Family corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a value of one in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
2. The Model corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed with a value of one in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.
3. The Brand ID corresponds to bits [7:0] of the EBX register after the CPUID instruction is executed with a value of one in the EAX register.

Cache and TLB descriptor parameters are provided in the EAX, EBX, ECX and EDX registers after the CPUID instruction is executed with a value of two in the EAX register. Please refer to the *APU-485 Intel® Processor Identification and the CPUID Instruction* (order number 241618) for further information on the CPUID instruction.

S-Spec	Core Stepping	Processor Signature	Core Freq (GHz)	Data Bus Freq (MHz)	L2 Cache Size	Package And Revision
SL6GV	C1	0F27h	1.60	400	512 K	42.5 mm FC-mPGA2

Errata

W1. Transaction is not Retried after BINIT#

Problem: If the first transaction of a locked sequence receives a HITM# and DEFER# during the snoop phase it should be retried and the locked sequence restarted. However, if BINIT# is also asserted during this transaction, the transaction will not be retried.

Implication: When this erratum occurs, locked transactions will not be retried.

Workaround: None identified.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W2. Invalid Opcode 0FFFh Requires a ModRM Byte

Problem: Some invalid opcodes require a ModRM byte and other following bytes, while others do not. The invalid opcode 0FFFh did not require a ModRM in previous generation microprocessors such as Intel® Pentium® II or Intel® Pentium III processors, but it is required in the Low Voltage Intel® Xeon™ processor.

Implication: The use of an invalid opcode 0FFFh without the ModRM byte may result in a page or limit fault on the Intel Xeon processor.

Workaround: To avoid this erratum use ModRM byte with invalid 0FFFh opcode.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W3. When in No-Fill Mode (CR0.CD=1) the Memory Type of Large (PSE-4M and PAE-2M) Pages are Wrongly Forced to Uncacheable

Problem: When the processor is operating in No-Fill Mode (CR0.CD=1), the page miss hardware incorrectly forces the memory type of large (PSE-4M and PAE-2M) pages to UC memory type regardless of the MTRR settings. By forcing the memory type of these pages to UC, load operations, which should hit valid data in the L1 cache, are forced to load the data from system memory. Some applications will lose the performance advantage associated with the caching permitted by other memory types.

Implication: This erratum may result in some performance degradation when using no-fill mode with large pages.

Workaround: None identified.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W4. Processor may Hang due to Speculative Page Walks to Non-Existent System Memory

Problem: A load operation issued speculatively by the processor that misses the Data Translation Lookaside Buffer (DTLB) results in a page walk. A branch instruction older than the load retires so that this load operation is now in the mispredicted branch path. Due to an internal boundary condition, in some instances the load is not canceled before the page walk is issued.

The Page Miss Handler (PMH) starts a speculative page-walk for the Load and issues a cacheable load of the Page Directory Entry (PDE). This PDE load returns data that point to a page table entry in uncacheable (UC) memory. The PMH issues the PTE Load to UC space, which is issued on the system bus. No response comes back for this load PTE operation since the address is pointing to system memory, which does not exist.

This load to non-existent system memory causes the processor to hang because other bus requests are queued up behind this UC PTE load, which never gets a response. If the load was accessing valid system memory, the speculative page-walk would successfully complete and the processor would continue to make forward progress.

Implication: Processor may hang due to speculative page walks to non-existent system memory.

Workaround: Page directories and page tables in UC memory space must point to system memory that exists.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W5. Memory Type of the Load Lock Different from its Corresponding Store Unlock

Problem: The Low Voltage Intel® Xeon™ processor employs a use-once protocol to ensure that a processor in a multiprocessor system may access data that is loaded into its cache on a Read-for-Ownership operation at least once before it is snooped out by another processor. This protocol is necessary to avoid a dual processor livelock scenario where no processor in the system can gain ownership of a line and modify it before that data is snooped out by another processor. In the case of this erratum, the use-once protocol incorrectly activates for split load lock instructions. A load lock operation accesses data that splits across a page boundary with both pages of WB memory type. The use-once protocol activates and the memory type for the split halves get forced to UC. Since use-once does not apply to stores, the store unlock instructions go out as WB memory type. The full sequence on the Bus is: locked partial read (UC), partial read (UC), partial write (WB), locked partial write (WB). The Use-once protocol should not be applied to Load locks.

Implication: When this erratum occurs, the memory type of the load lock will be different than the memory type of the store unlock operation. This behavior (Load Locks and Store Unlocks having different memory types) does not however introduce any functional failures such as system hangs or memory corruption.

Workaround: None identified.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W6. Machine Check Architecture Error Reporting and Recovery may not work as Expected

Problem: When the processor detects errors it should attempt to report and/or recover from the error. In the situations described below, the processor does not report and/or recover from the error(s) as intended.

- When a transaction is deferred during the snoop phase and subsequently receives a Hard Failure response, the transaction should be removed from the bus queue so that the processor may proceed. Instead, the transaction is not properly removed from the bus queue, the bus queue is blocked, and the processor will hang.
- When a hardware prefetch results in an uncorrectable tag error in the L2 cache, MC0_STATUS.UNCOR and MC0_STATUS.PCC are set but no Machine Check Exception (MCE) is signaled. No data loss or corruption occurs because the data being prefetched has not been used. If the data location with the uncorrectable tag error is subsequently accessed, an MCE will occur. However, upon this MCE, or any other subsequent MCE, the information for that error will not be logged because MC0_STATUS.UNCOR has already been set and the

MCA status registers will not contain information about the error which caused the MCE assertion but instead will contain information about the prefetch error event.

- When the reporting of errors is disabled for Machine Check Architecture (MCA) Bank 2 by setting all MC2_CTL register bits to 0, uncorrectable errors should be logged in the IA32_MC2_STATUS register but no machine-check exception should be generated. Uncorrectable loads on bank 2, which would normally be logged in the IA32_MC2_STATUS register, are not logged.
- When one half of a 64 byte instruction fetch from the L2 cache has an uncorrectable error and the other 32 byte half of the same fetch from the L2 cache has a correctable error, the processor will attempt to correct the correctable error but cannot proceed due to the uncorrectable error. When this occurs the processor will hang.
- When an L1 cache parity error occurs, the cache controller logic should write the physical address of the data memory location that produced that error into the IA32_MC1_ADDR REGISTER (MC1_ADDR). In some instances of a parity error on a load operation that hits the L1 cache, however, the cache controller logic may write the physical address from a subsequent load or store operation into the IA32_MC1_ADDR register.
- The local xAPIC has an Error Status Register, which records all errors it detects. Bit 6 of this register, the Receive Illegal Vector bit, is set when the local xAPIC detects an illegal vector in a message that it received. When an illegal vector error is received on the same internal clock that the error status register is being written due to a previous error, bit 6 does not get set and illegal vector errors are not flagged.
- When an error exists in the tag field of a cache line such that a request for ownership (RFO) issued by the processor hits multiple tag fields in the L2 cache (the correct tag and the tag with the error) and the accessed data also has a correctable error, the processor will correctly log the multiple tag match error but will hang when attempting to execute the machine check exception handler.
- If a memory access receives a machine check error on both 64 byte halves of a 128-byte L2 cache sector, the IA32_MC0_STATUS register records this event as multiple errors, i.e., the valid error bit and the overflow error bit are both set indicating that a machine check error occurred while the results of a previous error were in the error-reporting bank. The IA32_MC1_STATUS register should also record this event as multiple errors but instead records this event as only one correctable error.
- The overflow bit should be set to indicate when more than one error has occurred. The overflow bit being set indicates that more than one error has occurred. Because of this erratum, if any further errors occur, the MCA overflow bit will not be updated, thereby incorrectly indicating only one error has been received.
- If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the processor will signal a Machine Check Exception (MCE). If the instruction is directed at a device that is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, while attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is successfully completed, it will attempt to restart the I/O instruction, but will not have the correct machine state due to the call to the MCE handler. This can lead to failure of the restart and shutdown of the processor.
- If PWRGOOD is de-asserted during a RESET# assertion causing internal glitches, the MCA registers may latch invalid information.

- If RESET# is asserted, then de-asserted, and reasserted, before the processor has cleared the MCA registers, then the information in the MCA registers may not be reliable, regardless of the state or state transitions of PWRGOOD.
- If MCERR# is asserted by one processor and observed by another processor, the observing processor does not log the assertion of MCERR#. The Machine Check Exception (MCE) handler called upon assertion of MCERR# will not have any way to determine the cause of the MCE.
- The Overflow Error bit (bit 62) in the IA32_MC0_STATUS register indicates, when set, that a machine check error occurred while the results of a previous error were still in the error reporting bank (i.e. The Valid bit was set when the new error occurred). If an uncorrectable error is logged in the error-reporting bank and another error occurs, the overflow bit will not be set.
- The MCA Error Code field of the IA32_MC0_STATUS register gets written by a different mechanism than the rest of the register. For uncorrectable errors, the other fields in the IA32_MC0_STATUS register are only updated by the first error. Any further errors that are detected will update the MCA Error Code field without updating the rest of the register, thereby leaving the IA32_MC0_STATUS register with stale information.
- When a speculative load operation hits the L2 cache and receives a correctable error, the IA32_MC1_Status Register may be updated with incorrect information. The IA32_MC1_Status Register should not be updated for speculative loads.
- The processor should only log the address for L1 parity errors in the IA32_MC1_Status register if a valid address is available. If a valid address is not available, the Address Valid bit in the IA32_MC1_Status register should not be set. In instances where an L1 parity error occurs and the address is not available because the linear to physical address translation is not complete or an internal resource conflict has occurred, the Address Valid bit is incorrectly set.
- The processor may hang when an instruction code fetch receives a hard failure response from the system bus. This occurs because the bus control logic does not return data to the core, leaving the processor empty. IA32_MC0_STATUS MSR does indicate that a hard fail response occurred.
- The processor may hang when the following events occur and the machine check exception is enabled, CR4.MCE=1. A processor that has its STPCLK# pin asserted will internally enter the Stop Grant State and finally issue a Stop Grant Acknowledge special cycle to the bus. If an uncorrectable error is generated during the Stop Grant process it is possible for the Stop Grant special cycle to be issued to the bus before the processor vectors to the machine check handler. Once the chipset receives its last Stop Grant special cycle it is allowed to ignore any bus activity from the processors. As a result, processor accesses to the machine check handler may not be acknowledged, resulting in a processor hang.

Implication: The processor is unable to correctly report and/or recover from certain errors.

Workaround: None identified.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W7. Debug Mechanisms may not Function as Expected

Problem: Certain debug mechanisms may not function as expected on the processor. The cases are as follows:

- When the following conditions occur: 1) An FLD instruction signals a stack overflow or underflow, 2) the FLD instruction splits a page-boundary or a 64 byte cache line boundary, 3) the instruction matches a Debug Register on the high page or cache line respectively, and 4) the FLD has a stack fault and a memory fault on a split access, the processor will only signal the stack fault and the debug exception will not be taken.

- When a data breakpoint is set on the ninth and/or tenth byte(s) of a floating point store using the Extended Real data type, and an unmasked floating point exception occurs on the store, the break point will not be captured.
- When any instruction has multiple debug register matches, and any one of those debug registers is enabled in DR7, all of the matches should be reported in DR6 when the processor goes to the debug handler. This is not true during a REP instruction. As an example, during execution of a REP MOVSW instruction the first iteration a load matches DR0 and DR2 and sets DR6 as FFFF0FF5h. On a subsequent iteration of the instruction, a load matches only DR0. The DR6 register is expected to still contain FFFF0FF5h, but the processor will update DR6 to FFFF0FF1h.
- A Data breakpoint that is set on a load to uncacheable memory may be ignored due to an internal segment register access conflict. In this case the system will continue to execute instructions, bypassing the intended breakpoint. Avoiding having instructions that access segment descriptor registers e.g. LGDT, LIDT close to the UC load, and avoiding serialized instructions before the UC load will reduce the occurrence of this erratum.

Implication: Certain debug mechanisms do not function as expected on the processor.

Workaround: None identified.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W8. Cascading of Performance Counters does not work Correctly when Forced Overflow is Enabled

Problem: The performance counters are organized into pairs. When the CASCADE bit of the Counter Configuration Control Register (CCCR) is set, a counter that overflows will continue to count in the other counter of the pair. The FORCE_OVF bit forces the counters to overflow on every non-zero increment. When the FORCE_OVF bit is set, the counter overflow bit will be set but the counter no longer cascades.

Implication: The performance counters do not cascade when the FORCE_OVF bit is set.

Workaround: None identified

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W9. EMON Event Counting of x87 Loads may not work as Expected

Problem: If a performance counter is set to count x87 loads and floating-point exceptions are unmasked, the FPU Operand (Data) Pointer (FDP) may become corrupted.

Implication: When this erratum occurs, FPU Operand (Data) Pointer (FDP) may become corrupted.

Workaround: This erratum will not occur with floating point exceptions masked. If floating-point exceptions are unmasked, then performance counting of x87 loads should be disabled.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W10. System Bus Interrupt Messages without Data and which Receive a HardFailure Response may hang the Processor

Problem: When a system bus agent (processor or chipset) issues an interrupt transaction without data onto the system bus, and the transaction receives a HardFailure response, a potential processor hang can occur. The processor, which generates an inter-processor interrupt (IPI) that receives HardFailure response, will still log the MCA error event cause as HardFailure, even if the APIC causes a hang. Other processors, which are true targets of the IPI, will also hang on HardFailure-without-data, but will not record an MCA HardFailure event as a cause. If a HardFailure response occurs on a system bus interrupt message with data, the APIC will complete the operation so as not to hang the processor.

Implication: The processor may hang.

Workaround: None identified

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W11. Processor Flags #PF Instead of #AC on an Unlocked CMPXC8B Instruction

Problem: If a data page fault (#PF) and alignment check fault (#AC) both occur for an unlocked CMPXC8B instruction, then #PF will be flagged.

Implication: Software that depends on #AC before #PF will be affected since #PF is flagged in this case.

Workaround: Remove the software's dependency on the fact that #AC has precedence over #PF. Alternately, if the reload is due to a not present page, reload the page in the page fault handler and then restart the faulting instruction.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W12. FSW may not be Completely Restored after Page Fault on FRSTOR or FLDENV Instructions

Problem: If the FPU operating environment or FPU state (operating environment and register stack) being loaded by an FLDENV or FRSTOR instruction wraps around a 64 Kbyte or 4 Gbyte boundary and a page fault (#PF) or segment limit fault (#GP or #SS) occurs on the instruction near the wrap boundary, the upper byte of the FPU status word (FSW) might not be restored. If the fault handler does not restart program execution at the faulting instruction, stale data may exist in the FSW.

Implication: When this erratum occurs, stale data will exist in the FSW.

Workaround: Ensure that the FPU operating environment and FPU state do not cross 64 Kbyte or 4 Gbyte boundaries. Alternately, ensure that the page fault handler restarts program execution at the faulting instruction after correcting the paging problem.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W13. Processor Issues Inconsistent Transaction Size Attributes for Locked Operation

Problem: When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an eight byte load lock onto the system bus. A subsequent eight byte store unlock is expected, but instead a four byte store unlock occurs. Correct data is provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an eight byte store unlock.

Implication: No known commercially available chipsets are affected by this erratum.

Workaround: None identified.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W14. IA32_MC0_ADDR and IA32_MC0_MISC Registers will Contain Invalid or Stale Data Following a Data, Address, or Response Parity Error

Problem: If the processor experiences a data, address, or response parity error, the ADDRV and MISCV bits of the IA32_MC0_STATUS register are set, but the IA32_MC0_ADDR and IA32_MC0_MISC registers are not loaded with data regarding the error.

Implication: When this erratum occurs, the IA32_MC0_ADDR and IA32_MC0_MISC registers will contain invalid or stale data.

Workaround: Ignore any information in the IA32_MC0_ADDR and IA32_MC0_MISC registers after a data or response parity error.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W15. When the Processor is in the System Management Mode (SMM), Debug Registers may be Fully Writeable

Problem: When in System Management Mode (SMM), the processor executes code and stores data in the SMRAM space. When the processor is in this mode and writes are made to DR6 and DR7, the processor should block writes to the reserved bit locations. Due to this erratum, the processor may not block these writes. This may result in invalid data in the reserved bit locations.

Implication: Reserved bit locations within DR6 and DR7 may become invalid.

Workaround: Software may perform a read/modify/write when writing to DR6 and DR7 to ensure that the value in the reserved bits is maintained.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W16. Associated Counting Logic must be Configured when using Event Selection Control (ESCR) MSR

Problem: ESCR MSRs allow software to select specific events to be counted, with each ESCR usually associated with a pair of performance counters. ESCRs may also be used to qualify the detection of at-retirement events that support precise-event-based sampling (PEBS). A number of performance metrics that support PEBS require a second ESCR to tag micro-ops for the qualification of at-retirement events. (The first ESCR is required to program the at-retirement event.) Counting is enabled via counter configuration control registers (CCCR) while the event count is read from one of the associated counters. When counting logic is configured for the subset of at-retirement events that require a second ESCR to tag micro-ops, at least one of the CCCRs in the same group of the second ESCR must be enabled.

Implication: If no CCCR/counter is enabled in a given group, the ESCR in that group that is programmed for tagging micro-ops will have no effect. Hence a subset of performance metrics that require a second ESCR for tagging micro-ops may result in zero count.

Workaround: Ensure that at least one CCCR/counter in the same group as the tagging ESCR is enabled for those performance metrics that require two ESCRs and tagging micro-ops for at-retirement counting.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W17. Shutdown and IERR# may Result due to a Machine Check Exception on a Hyper-Threading Technology Enabled Processor

Problem: When a Machine Check Exception (MCE) occurs due to an internal error, both logical processors on a Hyper Threading technology enabled processor normally vector to the MCE handler. However, if one of the logical processors is in the “Wait for SIPI” state, that logical processor will not have a MCE handler and will shut down and assert IERR#.

Implication: A processor with a logical processor in the ‘Wait for SIPI’ state will shut down when an MCE occurs on the other thread.

Workaround: None identified.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W18. BPM[5:3]# V_{IL} does not meet Specification

Problem: The V_{IL} for BPM[5:3]# is specified as 0.9 * GTLREF [V]. Due to this erratum the V_{IL} for these signals is 0.9 * GTLREF - .100 [V].

Implication: The processor requires a lower input voltage than specified to recognize a low voltage on the BPM[5:3]# signals.

Workaround: When intending to drive the BPM[5:3]# signals low, ensure that the system provides a voltage lower than 0.9 * GTLREF - .100 [V].

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W19. Processor may hang under Certain Frequencies and 12.5% STPCLK# Duty Cycle

Problem: If a system de-asserts STPCLK# at a 12.5% duty cycle, and the processor is running below 2 GHz and the processor thermal control circuit (TCC) on-demand clock modulation is active, the processor may hang. This erratum does not occur under the automatic mode of the TCC.

Implication: When this erratum occurs, the processor will hang.

Workaround: If use of the on-demand mode of the processor's TCC is desired in conjunction with STPCLK# modulation, then assure that STPCLK# is not asserted at a 12.5% duty cycle.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W20. System may hang if a Fatal Cache Error causes Bus Write Line (BWL) Transaction to occur to the same Cache Line Address as an Outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)

Problem: A processor internal cache fatal data ECC error may cause the processor to issue a BWL transaction to the same cache line address as an outstanding BRL or BRIL. As it is not typical behavior for a single processor to have a BWL and a BRL/BRIL concurrently outstanding to the same address, this may represent an unexpected scenario to system logic within the chipset.

Implication: The processor may not be able to fully execute the machine check handler in response to the fatal cache error if system logic does not ensure forward progress on the system bus under this scenario.

Workaround: System logic should ensure completion of the outstanding transactions. Note that during recovery from a fatal data ECC error, memory image coherency of the BWL with respect to BRL/BRIL transactions is not important. Forward progress is the primary requirement.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W21. Re-mapping the APIC Base Address to a Value Less Than or Equal to 0xDC001000 may cause IO and Special Cycle Failure

Problem: Re-mapping the APIC base address from its default can cause conflicts with either I/O or special cycle bus transactions.

Implication: Either I/O or special cycle bus transactions can be redirected to the APIC, instead of appearing on the front-side bus.

Workaround: Use any APIC base addresses above 0xDC001000 as the relocation address.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W22. Erroneous BIST Result Found in EAX Register after Reset

Problem: The processor may show an erroneous BIST (built-in self test) result in the EAX register bit 0 after reset.

Implication: When this erratum occurs, an erroneous BIST failure will be reported in the EAX register bit 0, however this failure can be ignored since it is not accurate.

Workaround: It is possible for BIOS to workaround this issue by masking off bit 0 in the EAX register where BIST results are written.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W23. Simultaneous Assertion of A20M# and INIT# may Result in Incorrect Data Fetch

Problem: If A20M# and INIT# are simultaneously asserted by software, followed by a data access to the 0xFFFFFXXX memory region, with A20M# still asserted, incorrect data will be accessed. With A20M# asserted, an access to 0xFFFFFXXX should result in a load from physical address 0xFFEFFXXX. However, in the case of A20M# and INIT# being asserted together, the data load will actually be from the physical address 0xFFFFFXXX. Code accesses are not affected by this erratum.

Implication: Processor may fetch incorrect data, resulting in BIOS failure.

Workaround: Deasserting and reasserting A20M# prior to the data access will workaround this erratum.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W24. Processor does not Respond to Break Requests from ITP

Problem: On power-up and low-power state transitions, the processor's TAP circuitry may remain in the Tap-Logic-Reset (TLR) state.

Implication: The ITP is unable to cause a break on reset in the processor, which may prevent the loading of processor and chipset registers, or affect the ability to debug from cold boot and low power transitions.

Workaround: None identified.

Status: For the stepping affected see the *Summary Table of Changes* section of this document.

W25. A Write to APIC Task Priority Register (TPR) That Lowers Priority May Appear to Have Not Occurred

Problem: Uncacheable stores to the APIC space are handled in a non-synchronized way with respect to the speed at which instructions are retired. If an instruction that masks the interrupt flag, e.g., CLI, is executed soon after an uncacheable write to the TPR that lowers the APIC priority, the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR but higher than the final TPR to not be serviced until the interrupt flag is finally cleared, e.g., STI. Interrupts will remain pended and are not lost.

Implication: This condition may allow interrupts to be accepted by the processor but may delay their service.

Workaround: This can be avoided by issuing a TPR Read after a TPR Write that lowers the TPR value. This will force the store to the APIC priority resolution logic before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

Specification Changes

None for this revision of the Specification Update.

Specification Clarifications

W1. AE28 and AE29 Pins Signal Definitions are not VCC

The following change applies to the *Low Voltage Intel® Xeon™ Processor at 1.60 GHz Datasheet*:

The AE28 and AE29 pins signals are not truly “VCC”. These signals should be "VID_VCC". VID_VCC is required for correct operation of the Low Voltage Intel® Xeon™ processor VID and BSEL logic.

Documentation Changes

W1. Inaccurate Operation Description for Shift Instructions

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Section 3.2 "Instruction Reference" Operation description for the SAL/SAR/SHL/SHR instruction is incorrect. It currently states:

```
(* Determine overflow for the various instructions *)
IF COUNT = 1
    THEN
            IF instruction is SAL or SHL
…
ELSE IF COUNT = 0
    THEN
            All flags remain unchanged;
```

It should state:

```
(* Determine overflow for the various instructions *)
IF (COUNT AND 1FH) = 1
    THEN
            IF instruction is SAL or SHL
…
ELSE IF (COUNT AND 1FH) = 0
    THEN
            All flags remain unchanged;
```

W2. Documentation Changes to Support the Use of Hyper-Threading Technology

The *Intel Architecture Software Developer's Manual, Vol 1: Basic Architecture* Chapter 2, and *Vol 3: System Programming Guide* Chapter 7:

Both chapters have been rewritten to include Hyper-Threading Technology terminology and descriptions.

W3. IRET/IRETD TASK-RETURN Block has Incorrect Exception Information

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Chapter 3, Section 3.2 "Instruction Reference" incorrectly describes the TASK-RETURN block of the OPERATION section of the IRET/IRETD instruction. It currently states:

IF local/global bit is set to local
OR index not within GDT limits
THEN #GP(TSS selector);
…
IF TSS descriptor type is not TSS or if the TSS is marked not busy
THEN #GP(TSS selector);

It should state:

IF local/global bit is set to local
OR index not within GDT limits
THEN #TS(TSS selector);
…
IF TSS descriptor type is not TSS or if the TSS is marked not busy
THEN #TS(TSS selector);

W4. LGDT Exception Listing in Virtual-8086

The Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference Chapter 3, Section 3.2 "Instruction Reference" incomplete listing of LGDT/LIDT in Virtual-8086 Mode Exceptions. It currently states:

Virtual-8086 Mode Exceptions

#GP(0) The LGDT and LIDT instructions are not recognized in virtual-8086 mode.

It should state:

Virtual-8086 Mode Exceptions

#GP(0) The LGDT and LIDT instructions are not recognized in virtual-8086 mode.

#GP If the current privilege level is not 0.

W5. Various Table B-22 Errors and Omissions

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* APPENDIX B, Section B.7 "Floating-point Instruction Formats and Encoding" various errors and omission in Table B-22. It currently states:

FICOMP-Compare Integer and Pop

16-bit memory

32-bit memory

FIDIV

ST(0) ⇓ ST(0) + 16-bit memory

ST(0) ⇓ ST(0) + 32-bit memory

FIDIVR

ST(0) ⇓ ST(0) + 16-bit memory

ST(0) ⇓ ST(0) + 32-bit memory

.
.
.
FIMUL

ST(0) ⇓ ST(0) + 16-bit memory
ST(0) ⇓ ST(0) + 32-bit memory

.
.
.
FISUB

ST(0) ⇓ ST(0) + 16-bit memory
ST(0) ⇓ ST(0) + 32-bit memory

FISUBR

ST(0) ⇓ ST(0) + 16-bit memory
ST(0) ⇓ ST(0) + 32-bit memory

.
.
.
FMULP – Multiply

ST(0) ⇓ ST(0) X ST(i)

.
.
.

It should state:

FICOMP-Compare Integer and Pop

16-bit memory
32-bit memory

FIDIV

ST(0) ⇓ ST(0) ÷ 16-bit memory
ST(0) ⇓ ST(0) ÷ 32-bit memory

FIDIVR

ST(0) ⇓ 16-bit memory ÷ ST(0)
ST(0) ⇓ 32-bit memory ÷ ST(0)

.
.
.
FIMUL

ST(0) ⇓ ST(0) × 16-bit memory
ST(0) ⇓ ST(0) × 32-bit memory

.
.
.

FISUB

ST(0) ⇓ ST(0) - 16-bit memory

ST(0) ⇓ ST(0) - 32-bit memory

FISUBR

ST(0) ⇓ 16-bit memory - ST(0)

ST(0) ⇓ 32-bit memory - ST(0)

.
.
.

FMULP – Multiply

ST(i) ⇓ ST(0) × ST(i)

.
.
.

W6. Misreporting of Exceptions for MOVDQA

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Section 3.2, Instruction Reference, under MOVDQA currently states:

Protected Mode Exceptions

#GP(0)	If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.
	If memory operand is not aligned on a 16-byte boundary, regardless of segment.
#SS(0)	If a memory operand effective address is outside the SS segment limit.
#NM	If TS in CR0 is set.
#UD	If EM in CR0 is set.
	If OSFXSR in CR4 is 0.

Real-Address Mode Exceptions

#PF(fault-code)	If a page fault occurs.
#GP(0)	If memory operand is not aligned on a 16-byte boundary, regardless of segment.
	If any part of the operand lies outside of the effective address space from 0 to FFFFH.
#NM	If TS in CR0 is set.
#UD	If EM in CR0 is set.
	If OSFXSR in CR4 is 0.

It should state:

Protected Mode Exceptions

#PF(fault-code)	If a page fault occurs.
#GP(0)	If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.

	If memory operand is not aligned on a 16-byte boundary, regardless of segment.
#SS(0)	If a memory operand effective address is outside the SS segment limit.
#NM	If TS in CR0 is set.
#UD	If EM in CR0 is set.
	If OSFXSR in CR4 is 0.

Real-Address Mode Exceptions

#GP(0)	If memory operand is not aligned on a 16-byte boundary, regardless of segment.
	If any part of the operand lies outside of the effective address space from 0 to FFFFH.
#NM	If TS in CR0 is set.
#UD	If EM in CR0 is set.
	If OSFXSR in CR4 is 0.

W7. Effect of STI on NMI

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Section 3.2, Instruction Reference, under STI-Set Interrupt Flag currently states:

> The IF flag and the STI and CLI instructions have no affect on the generation of exceptions and NMI interrupts.

It should state:

> The IF flag and the STI and CLI instructions have no affect on the generation of exceptions. NMI interrupts may be blocked for one macroinstruction following an STI.

W8. 133 MHz PSB Encoding in MSR_EBC_FREQUENCY_ID Register

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Appendix B, Table B-1, the MSR_EBC_FREQUENCY_ID register bits[18:16] currently states:

2CH	44	MSR_EBC_ FREQUENCY_ID	Shared	Processor Frequency Configuration. (R) Indicates current processor frequency configuration.
		20:0		**Reserved.**
		18:16		**Scalable Bus Speed.** (R/W) Indicates the intended scalable bus speed: Encoding Scalable Bus Speed 000B 100 MHz All Others Reserved
		63:24		**Reserved.**

It should state:

2CH	44	MSR_EBC_ FREQUENCY_ID	Shared	Processor Frequency Configuration. The bit field layout of this MSR varies according to the MODEL value of the CPUID version information. The following bit field layout applies to Pentium 4 and Xeon Processors with MODEL encoding equal or greater than 2. (R) Indicates current processor frequency configuration.
		15:0		**Reserved.**
		18:16		**Scalable Bus Speed.** (R/W) Indicates the intended scalable bus speed: <u>Encoding</u> <u>Scalable Bus Speed</u> 000B 100 MHz 001B 133 MHz (The actual value of 133.33MHz should be utilized if performing calculation with System Bus Speed, when encoding is 001B.) All Others Reserved
		63:29		**Reserved.**

W9. Double Fault Exception

The Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide Chapter 5, Section 5.12 "Exception and Interrupt Reference" Interrupt 8 - Double Fault Exception (#DF), the double fault handler description currently states:

> If the shutdown occurs while the processor is executing an NMI interrupt handler, then only a hardware reset can restart the processor.

It should state:

> If the shutdown occurs while the processor is executing an NMI interrupt handler, then only a hardware reset can restart the processor. Likewise, if the shutdown occurs while executing in SMM, a hardware reset should be used to restart the processor.

W10. Move To/From Control Registers

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Section 3.2 "Instruction Reference" Operation description for the Move to/from Control Register instruction is incorrect. It currently states:

> When loading a control register, a program should not attempt to change any of the reserved bits; that is, always set reserved bits to the value previously read.

It should state:

> If a load of CR4 would set any of its reserved bits, a general-protection fault occurs. The reserved bits in CR0 and CR3 remain clear after any load of those registers, although attempts to set such bits do not cause faults. On Intel Pentium 4, Intel Xeon, and P6 family processors, CR0.ET remains set after any load of CR0, although attempts to clear it do not cause faults. When loading a control register, a program should always set reserved bits to the value previously read.